维普资讯 http://www.cqvip.com

文章编号:1007-1385(2004)02-0086-03

图论中最短路径问题的解法

项荣武　刘艳杰　胡忠盛
(沈阳药科大学基础学院,辽宁 沈阳 110016)

摘　要:图论在解决运筹学、网络理论、控制论等领域的问题中显示出优越性。特别是最短路径问题被广泛的应用在工程、运输等方面,尤其在运筹学模型中最短路径问题已是不可缺少的一种方法。因此就其运算解法进行了编译,并用VC++、Matlab两种程序完成其算法以寻求较快捷的解法。

关键词:最短路径算法;图论;运筹学
中图分类号:O157.6　　**文献标识码**:A

本文对图论中的最短路径问题进行分析,就其运算解法进行了编译,以寻求较快捷的解法。而关于它的解法自然也受到同行业的注视,现就其解法从两种情况进行介绍:

1　某顶点到其余顶点最短路径问题

给定一个有向图 $G=(V,E,W)$ 以及某个顶点 $v_0\in V(G)$,求到其他各顶点的最短路径的算法叫做Dijkstra算法。

在研究从某源点到其余顶点最短路径的问题时有以下性质:(1)假设 S 是唯一求得最短路径的终点的集合(S 的初态为空集),则下一条长度较长的最短路径(设它的终点为 x)或者是弧 $\langle v_0,x\rangle$;或者是中间只经过 S 集合中的顶点,最后到达顶点 x 的路径。

利用此性质(2),按路径长度不减的顺序产生最终路径。为此,引进一个辅助向量dist。对于不在 S 中的顶点 w,令 dist(w)表示从 v_0 开始,且通过 S 中的顶点到达每个终点 w 的最短路径的长度。若从 v_0 到 w 有路径,则 dist(w)为边的权值;若从 v_0 到 w 没有路径,则 dist(w)为∞。于是所产生的下一条最短路径的终点,必定是从 v_0 到所有的那些不在 S 中,且路径的长度为最小的顶点。也就是说,如果假设下一条最短路径的终点是 u,必有

$$\mathrm{dist}(u)=\min\{\mathrm{dist}(w)\mid w\notin S, w\in V(G)\}$$

由于确定了 v_0-u 下一条最短路径,于是把顶点 u 加进集合 S,从 v_0 开始到(其中间点在 S 中,而终点 w 不在 S 中)的那些路径,因为选择了 u,u 便在 S 中了,于是这条路径可以从 v_0 经过 u 到 w,而且完全有可能从 v_0 经过 u 到达 w,比从 v_0 不经过 u 到达 w 的路径要短,这时要修改 dist(w),即若

$$\mathrm{dist}(u)+\mathrm{cost}(u,w)<\mathrm{dist}(w)$$

则修改 dist(w)为

$$\mathrm{dist}(w)=\mathrm{dist}(u)+\mathrm{cost}(u,w)$$

注:cost表示图的邻接矩阵,对有向图的顶点从1至 n 编号。

根据以上分析最短路径算法,现在分别介绍用VC++、Matlab编写程序实现其算法:

(1) 在VC++环境中编译的程序如下[1]:(有权无向图)

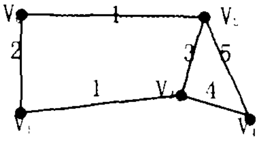
```
# include 〈iostream.h〉
# define MAX 50
# define up 50000
int cost[MAX][MAX]; int dist[MAX],n;
struct
{
  int num; int pnode[MAX];
} path[MAX];
void creatgraph()
{
  int i, j, s, p, l, contin = 1;
  cout 〈〈"顶点个数:";   cin 〉〉n;
  for (i = 0;i < n;i + + )
  {
    for (j = 0;j < n;j + + )
      cost[i][j] = cost[j][i] = up; cost[i][i] = 0;
```

收稿日期:2003-09-30
作者简介:项荣武(1977-),男,辽宁省锦州人,助教

维普资讯 http://www.cqvip.com


```
}
......}
void shortdjs()
{
int s[MAX]; int mindis, dis, i, j, V0 = 0, u;
for (i + 0;i < n;i + + )
{
  dist[i] = cost[V0][i]; path[i].pnode[0] = V0;
path[i].num = 0;
  s[i] = 0;
  }
  s[V0] = 1;
  for (i = 1;i < n;i + + )
  {
    ......
}
void dispath()
{
  int i, j;
  cout <<" \ n 从 V0 到各顶点的最短路径长度如下: \ n";
  cout << " \ t(起点 - >终点)  最短长度  最短路径 \ n";
  cout <<" \ t- - - - - - - - -   - - - - - - - - -   - - - - - - - - - \ n";
  for(i = 1;i < n;i + + )
  {
  ......
    }
void main()
{
  creatgraph(); shortdjs(); dispath();
}
```

%例 1



用 VC + + 程序执行结果如下:

从 V_0 到各顶点的最短路径长度如下:

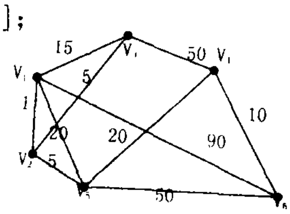
起点→终点	最短长度	最短路径
$V_0 \to V_1$	2	V_0, V_1
$V_0 \to V_2$	1	V_0, V_2
$V_0 \to V_3$	3	V_0, V_1, V_3
$V_0 \to V_4$	6	V_0, V_2, V_4

(2)在 Matlab 环境中编译程序如(3):(有权有向图)

建立 minroad.M 函数

```
function  [P,S(R)] = minroad(i, m, W)
%  i 为最短路径的起始点, m 为图顶点数, W 为图的带权邻接矩阵,
%  不构成边的两顶点之间的权用 inf 表示。
%  S 是矩阵, S 的每一列从上到下记录了从始点到终点的最短路径所经顶点的序号;
%  R 是一行向量,记录了 S 中所示路径的大小;
%  P 是矩阵, P 中的每一列从上到倒数第二行记录了从始点到终点的最短路径所经顶点的序号;最后一行记录了该列所示路径的大小;
%  S(R)的意思是:该位置根据需要书写 S 或 R。
  d = [];pp = []; n = []; n(1,1) = i; V = 1:m; V(i) = []; d = [0;i];
%  d 的第二行是每次求出的最短路径的终点,第一行是最短路径的值
    p = 2;[a,c] = size(d);
while ~ isempty(V)
  [td,j] = min(W(i,V));tj = V(j);
  for k = 2:c
    [t1,jj] = min(d(1,k) + W(d(2,k),V));
    t2 = V(jj);pp(k - 1,:) = [t1,t2,jj];
  end
  t = [td,tj,j;pp];[t3,t4] = min(t(:,1));
  if t3 = = td, n(1:2,p) = [i;t(t4,2)];
  else t5 = find(n(:,t4) ~ = 0);t6 = length(t5);
    if d(2,t4) = = n(t6,t4)
      n(1:t6 + 1,p) = [n(t5,t4);t(t4,2)];
      else,n(1:3,p) = [i;d(2,t4);t(t4,2)];
  end;end
  d = [d,[t3;t(t4,2)]];V(t(t4,3)) = [];
  [a,c] = size(d);p = p + 1;
end; S = n; R = d(1,:);
  P = [S;R];
```

%例 2



用 Matlab 执行运算过程:

≫ $w = inf * ones(6)$; $w(1,3) = 15$; $w(1,5) = 20$; $w(1,6) = 90$; $w(2,3) = 5$; $w(3,4) = 50$; w

维普资讯 http://www.cqvip.com

(4,6) = 10;

≫ $w(5,4)=20$; $w(5,6)=50$; $w(1,2)=1$; $w(2,5)=5$; $i=1$; $P=\text{minroad}(i,6,w)$

执行结果:$P=$

1	1	1	1	1	1
0	2	2	2	2	2
0	0	3	5	5	5
0	0	0	0	4	4
0	0	0	0	0	6
	0	1	6	6	26

2 每一对顶点间的最短路径问题

每次以一个顶点为起点,重复执行算法 n 次,可求得每一对顶点之间的最短路径。

求最短路径的 Floyed 算法,相对较简单。从有向图的带树的邻接矩阵 cos$t$ 出发,对有向图的 n 个顶点编号。若从 i 到 j 有弧($i=1,2,\Lambda,n,j=1,2,\Lambda,n$),则从 i 到 j 存在一条长度为 cos$t(i,j)$的路线。但该路径不一定是最短路径,尚需修改,修改的方法是进行 n 次试探。首先,考虑路径$(i,1,j)$(即在 i,j 中插进点 1),看$\langle i,1\rangle$,$\langle 1,j\rangle$是否存在,若存在,再比较(i,j)与$(i,1,j)$这两条路径,取长度较短者为当前求得的最短路径。于是这条求得的最短路径的中间点序号不大于 1。再在各对点 i,j 中插进一个点 2,看 $<i,\cdots,2>$,$<2,\cdots,j>$是否存在,若不存在,那么当前的最短路径仍是上次求得的中间点序号不大于 1 的最短路径;若存在,则将$(i,\Lambda,2,\Lambda,j)$的路径与前次求得的中间点序号不大于 1 的最短路径进行比较,取长度较短者为当前最短路径。这样,这次求得的最短路径的中间点序号不大于 2……依次类推,直至求得从 i 到 j 的最短路径。

为采用邻接矩阵来表示有向图,且每次修改都是在 n 阶矩阵上进行,首先定义一个矩阵序列:

$$A^{(0)},A^{(1)},\Lambda,A^{(k)},\Lambda,A^{(n)}$$

其中,$A^{(0)}=\text{cost}(i,j)$,$A^{(k)}$是从 i 到 j 的中间点序号不大于 k 的最短路径的长度,$A^{(k)}$ 的递推公式为:$A^{(k)}=\min\{(A^{(k-1)}(i,j)),A^{(k-1)},(i,k)+A^{(k-1)}(k,j)\}(1\le k\le n)$

【算法分析】[4] 最短路径(cos$t$,$S(i,j)$,$R(i,j)$)

cos$t$ 为有向图的邻接矩阵,有向图的顶点从 1 到 n 编号,S 和 R 均为 n 阶方阵,$S(i,j)$为从 i 到 j 的最短路径长度,$R(i,j)$为相应的路径

循环 i 以 1 为步长 从 1 至 n 执行
  [$S(i,j)\leftarrow$cos$t$ (i,j)
   若 $i\neq j$ 且 $S(i,j)<\max$
 则 $R(i,j)\leftarrow(i)+(j)$
  循环 k 以 1 为步长 从 1 至 n 执行
   循环 i 以 1 为步长 从 1 至 n 执行
    循环 j 以 1 为步长 从 1 至 n 执行
     若 $S(i,j)+S(i,j)<S(i,j)$
     则 [$S(i,j)\leftarrow S(i,k)+S(k,j)$
      $R(i,j)\leftarrow R(i,k)+R(k,j)$]
      {算法结束}

该程序的编译略。

上述对最短路径问题的运算程序分析及编译,在大量案例运算中相对于同类其他编程,执行速度比较快捷、准确。该程序是值得推广的。

参考文献:

[1]何渝.计算机常用数值算法与程序(C++版)[M].北京:人民邮电出版社,2003. 112~114

[2]李春葆,曾慧,张植民.数据结构程序设计题典[M].北京:清华大学出版社,2002. 65~68

[3]苏晓生,掌握 MATLAB6.0 及其工程应用[M].北京:科学出版社,2002.123~124

[4]康晓东.数值算法与非数值算法[M].北京:电子工业出版社,2003.134~135

Solution of the shortcut problem in graph algorithms

XIANG Rong - wu LIU Yan - jie HU Zhong - sheng

(The Department of Basic Courses, Shenyang Pharmaceutical University, Liaoning Shenyang 110016)

Abstract: Graph theory was a new subject which was rising quickly and was applied widely in recent years. It had advantages in solving the problems in many fields such as operational research, net theory and controlling theory. This article analyzes the shortest path problem of graph theory which come forth in engineering and transportation, especially about operational research model. So that by algorithm the shortest path resolution software was compiled. Its algorithm was completed with the two Languages, VC++ and Matlab, to seek the quicker algorithm.

Keywords: shortest path algorithm; graph theory; operational research